How AGL tests its distro and what challenges we face

Jan-Simon Möller
Automotive Grade Linux

/* Introduction */

The Automotive Grade Linux project (AGL) has setup a extensive test infrastructure and does use it not only for its releases but also during the daily development. This talk wil show what components are used and discusses the challenges that exist and what future development might be needed.

Automotive Grade Linux

Automotive Grade Linux

- 'Code first' project
- Goal is to create platform for use of Linux in automotive
- Multiple use-cases supported

AUTOMOTIVE GRADE LINUX

the only organization addressing all software in the car

Infotainment

Instrument Cluster

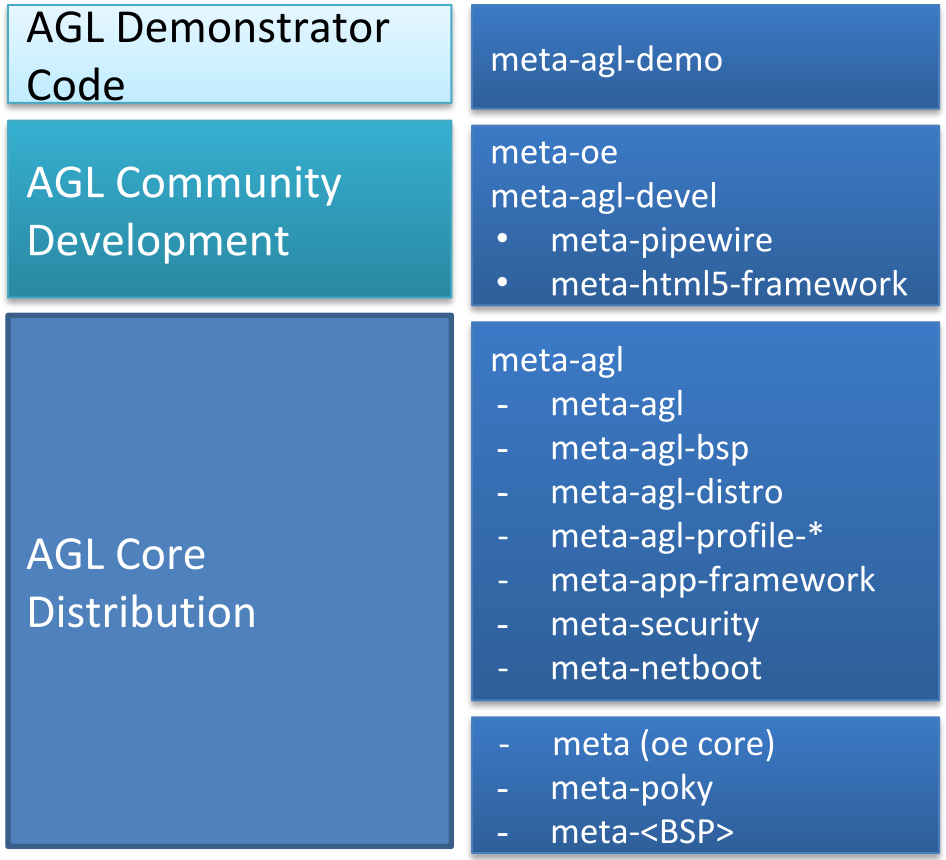
Heads-up Display (HUD)

Telematics/ Connectivity

Functional Safety

Advanced Driver Assistance Systems (ADAS)

- Single Sign on Using LF Identity
- Yocto based Build System
- Git repositories with Gerrit code review
  - https://gerrit.automotivelinux.org
- JIRA used for Project Management and Bug Tracking
  - https://jira.automotivelinux.org
- Jenkins Continuous Integration
  - All code changes built by Jenkins prior to merging
  - Daily snapshot builds https://download.automotivelinux.org/AGL/snapshots/master/



Testing challenges for AGL

x*y*z

We have:

- A core platform and multiple verticals on-top
  - Multiple final images (one or more per vertical)
- Multiple hardware platforms to run on
  - Renesas R-Car Gen3, x86-64, …. , Pi4
  - thus aarch64, x86-64, arm32

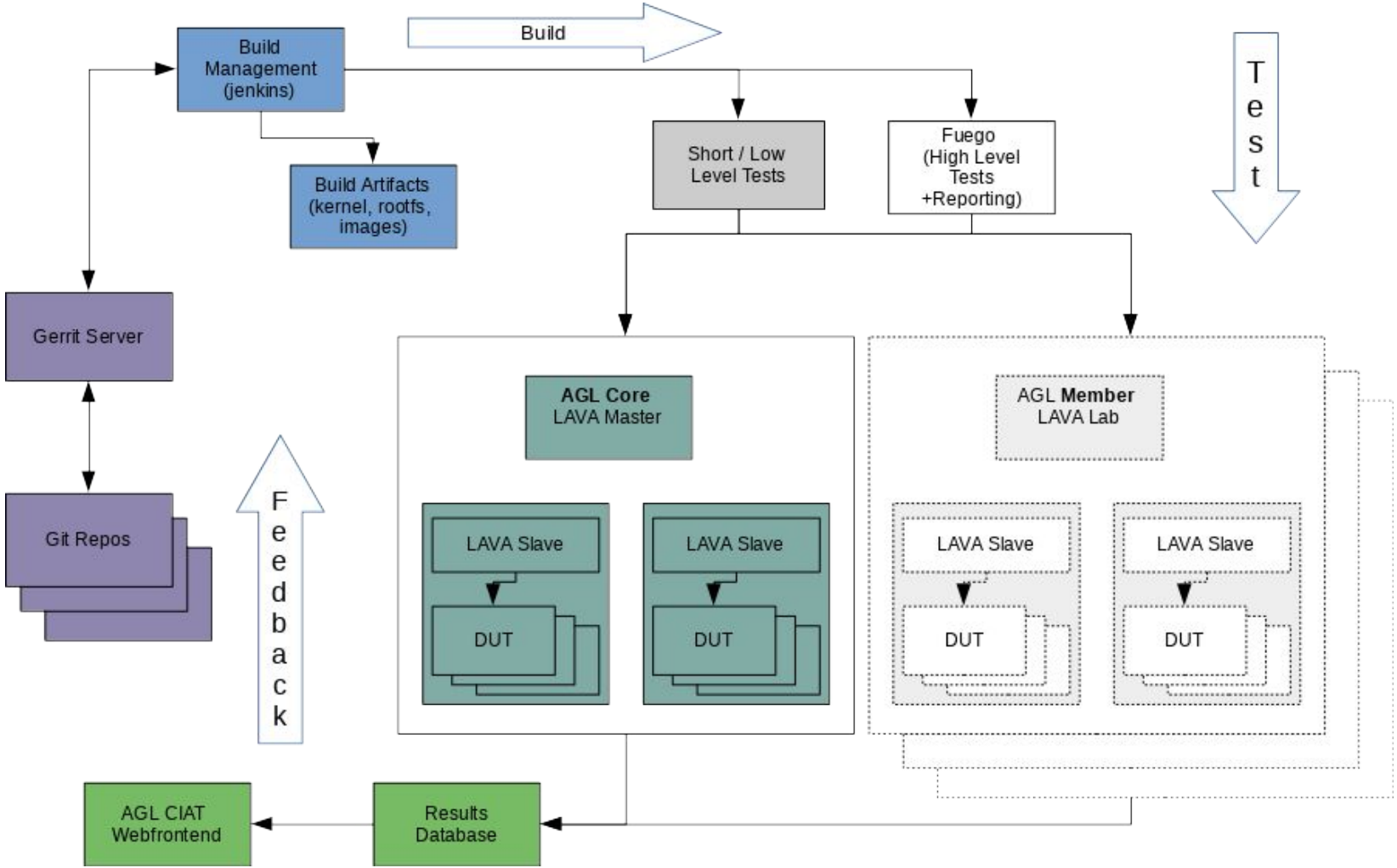
Timing

- Need results in a timely manner (CI) vs. complete coverage for releases

- CI for patches should give us a +1 back ASAP (aka hours, not days)
  - Need to ensure good turnaround for developers

- Release builds need a full pass and do things like
  - license scanning, cve scanning, ...

How do things look like in AGL ?


Build
Build Management (jenkins)
Build Artifacts (kernel, rootfs, images)
Short / Low Level Tests
Fuego (High Level Tests +Reporting)
Test
Gerrit Server
Git Repos
Feedback
AGL Core LAVA Master
LAVA Slave
DUT
LAVA Slave
DUT
AGL Member LAVA Lab
LAVA Slave
DUT
LAVA Slave
DUT
AGL CIAT Webfrontend
Results Database

Lessons learned (build)

- Full builds take considerable time
  - 5-6h on 16-core cloud machine
  - I/O an issue in cloud environment (reads/writes might trigger multiple actions in backend)
- Builds with yocto sstate-cache
  - shortest on x86-64 (15min)
  - longest on R-Car3 (80 min)
- → Use touchstone builds
- Cloud host spin-up time >~5min !

Lessons learned (test)

- Download speed to distributed labs becomes an issue
  - not just testing kernel image and 'same' userspace
  - userspace is ~800MB compressed !
- Build jobs tend to finish in bursts
  - multiple test jobs triggered at once
  - thus multiple downloads to the same lab at once

LAVA:

- Provide way to priorize machines in one lab over another when scheduling (currently we tend to pick always a particular lab)
- → better load-balancing across multiple workers with same set of HW attached

Lessons learned (reporting)

- KernelCI UI (current one) too slow for useful reporting of userspace tasks
- Just limited history in UI (aka 14 days) - we'd complete history

Q/A ?!